Formal Analysis of Security Protocols

Shruti Biswal

Towards the fulfillment of AERE/COMS 507X course requirements
Dr. Kristin Yvonne Rozier

The aim of this project is to represent verification of security protocols by use of a symbolic model checker, NuXMV. The project illustrates the formal analysis of security protocol by describing three different well-known protocol models namely Needham-Schroder Public Key Protocol, Otway-Rees Protocol and Kerberos Protocol. Analysis of a protocol is targeted towards detection of attacks in the protocol and suggestive modifications to the protocol that can eradicate the attack detected. In larger perspective, the project forms the basis of development and re-development of "secure" security protocols.

1 Introduction

Security protocols provision the authentication procedure of message passing among authorized users while ensuring non-transmission of information to any unauthorized agents. While the protocols are designed to ensure the same, it is equally important to find scenarios where the protocol fails. Formal analysis is one such method to detect a possible attack on the protocol. While detection of an attack on a protocol can be termed as a gap in the security protocol, recognition of the nature of attack encourages towards development of a more secure and convoluted protocol aimed towards avoidance of the attacks seen so far.

The project presents the formal analysis of three protocols in an order where each protocol is an improvement over the previous. Despite the fact of inter-improvement between protocols, each protocol has a possible attack detected, though each attack is of different kind. Detection of such attacks is simplified by use of model checkers that can trace paths where the security specification fails to hold.[**?**]

The remainder of the report is organized as follows. Section 2 describes the protocols relevant to this project. Section 3 discusses the components and the design used to model each protocol in NuXMV. Section 4 provides the implementation details of project. Section 5 comprises the results obtained from the execution of the models and discusses the nature of attack on the protocols that were found. Finally, conclusion is drawn in Section 6.

2 Background Details[?]

In order to model a system, it is important to identify the key features which need to be drawn and the other less important attributes that need to be abstracted. This section describes the important stages and components involved in the protocols and their execution. The details from this section are used to develop the models of the protocols in NuXMV in the subsequent sections.

2.1 Needham-Schroeder Public Key Protocol

Needham-Schroeder Public-Key Protocol aims at providing mutual authentication between two agents that communicate on a network via public keys. The two agents, A and B involved in the communication have public keys K_A and K_B that are available to the public for encryption purpose only.

Both agents A and B are known to generate nonces, which are unique numbers used for authentication purpose during session setup by the generating agents.

Table 1: Symbols used in Needham-Schroeder Public Key Protocol

A	Identifier of agent A
B	Identifier of agent B
N_A	Unique random number generated by agent A
N_B	Unique random number generated by agent B
K_A	Public key of A
K_B	Public key of B

The message exchanges that take place in the protocol are as follows :

1. $A \longrightarrow B : \{N_A, A\}_{K_B}$
   A sends B its nonce and identity. B decrypts the message and confirms the identification of the sender, A.

2. $B \longrightarrow A : \{N_A, N_B\}_{K_A}$
   After B sends its nonce N_B together with N_A, A decrypts the message and matches its sent N_A with the received N_A and authenticates B as an honest receiver.

3. $A \longrightarrow B : \{N_B\}_{K_B}$
   After A the nonce N_B to$B$, B decrypts the message and matches its sent N_B with the received N_B and authenticates A as an honest sender.

2.2 Otway-Rees Protocol

Otway-Rees Protocol distributes sessions keys generated by a server to the honest agents to allow safe exchange of messages between them. The session keys sent to the honest agents is always sent scrambled to avoid interception by intruders. The honest agents A and B request the server S for their session key, where B is the point of interaction with the server, S.

The message exchanges that take place in the protocol are as follows :

1. $A \longrightarrow B : A, B, \{N_A, Sn, A, B\}_{K_{AS}}$
   A sends relevant information to S via B in order to receive a session key to use for secure exchange of information with B later.

Table 2: Symbols used in Otway-Rees Protocol

A	Identifier of agent A
B	Identifier of agent B
S	Server's name
Sn	Session number
N_A	Unique random number generated by agent A
N_B	Unique random number generated by agent B
K_{AS}	Symmetric key shared between A and S
K_{BS}	Symmetric key shared between B and S
K_{AB}	Symmetric key shared between A and B

2. $B \longrightarrow S : A, B, \{N_A, Sn, A, B\}_{K_{AS}}, \{N_B, Sn, A, B\}_{K_{BS}}$
   B generates its own message packet and relays the received message from A to server, S.

3. $S \longrightarrow B : \{N_A, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}}$
   S decrypts both message packets to verify the session number, Sn in which the honest agents want to communicate and to verify their identities using A and B. S then generates the session key and sends it along with the respective nonces for verification purpose.

4. $B \longrightarrow A : \{N_A, K_{AB}\}_{K_{AS}}$
   B decrypts the packet encrypted with K_{BS} and accepts the session key after verifying its nonce, N_B. B then relays the other packet to A and A accepts the session key after verifying its nonce, N_A.

2.3 Kerberos Protocol

The Kerberos protocol authenticates by using shared secrets. The protocol verifies the identity of the agents by confirming that both agents have the same piece of secret information (session key). The agents authenticate themselves with the server by sending their identifiers and the server sends session keys with lifetime. The lifetime of the session keys disallows a session to last forever. Timestamp plays a significant role in authenticating and determining the validity of the session key. It is used to ensure that the session key is recent.

The message exchanges that take place in the protocol are as follows :

1. $A \longrightarrow S : \{A, B\}$
   A sends the its own identifier and the identifier of the agent with whom

Table 3: Symbols used in Kerberos Protocol

A	Identifier of agent A
B	Identifier of agent B
S	Server's name
T_S	Local timestamp of server, S
T_A	Local timestamp of agent, A
L	Lifetime of the ticket, K_{AB}
K_{AS}	Symmetric key shared between A and S
K_{BS}	Symmetric key shared between B and S
K_{AB}	Symmetric key shared between A and B, also called ticket

it wants to establish a connection. S verifies the agents.

2. $S \longrightarrow A : \{K_{AB}, T_S, L, B\}_{K_{AS}}, \{K_{AB}, T_S, L, A\}_{K_{BS}}$
   S generates a session keys and sends two copies of the session keys together with its local time and lifetime of the key encrypted with the respective symmetric keys of both agents.

3. $A \longrightarrow B : \{T_A, A\}_{K_{AB}}, \{K_{AB}, T_S, L, A\}_{K_{BS}}$
   A decrypts its own copy, creates a packet containing its local time & identifier encrypted with the session key. A also relays the other packet as it is to the the agent B with whom it intends to establish a safe communication.

4. $B \longrightarrow A : \{T_A + 1\}_{K_{AB}}$
   B decrypts the packet which was sent by the server originally and extracts the session key , K_{AB} to use it to decrypt the other packet generated by A to extract the timestamp. B sends the increased timestamp encrypted with the session key , so that A could verify that B indeed had the same session key as A.

3 Analysis and Design

This section describes how each protocol is modelled in NuXMV. The design includes representation of the components and abstraction of the complex features of a protocol while preserving the major functionality. Analysis refers to establishment of connection between the protocol and the design in terms of the specifications. Model validation in the protocol refers to allowance of undesirable communications between the components; however they do not lead to successful completion of the transaction.

3.1 Needham-Schroeder Public Key Protocol

3.1.1 Model

The NuXMV model includes the following modules to represent the different components of Needham-Schroeder Public Key Protocol:

- `module alice` : To represent agent A
- `module bob` : To represent agent B

Each message packet traversing in the channel includes 2 fields encrypted using a certain key. The parameters to a module include two such packets one for sending, the other for receiving. Each of the parameters are allowed to hold certain values pertaining to the protocol execution. The parameters that each module take are :

- `send1`, `send2` , `sendkey`: To represent packet sent by an agent.
- `rcv1`, `rcv2` , `rcvkey` : To represent packet received by an agent.

Each agent (or module) also have a set of states that refer to the stage in which the agent is in during the execution of the protocol :

- For agent A : {`a_initiate,a_wait,a_rcv_and_send,a_done`}
- For agent B : {`b_free,b_receive,b_checks_sender,b_done`}

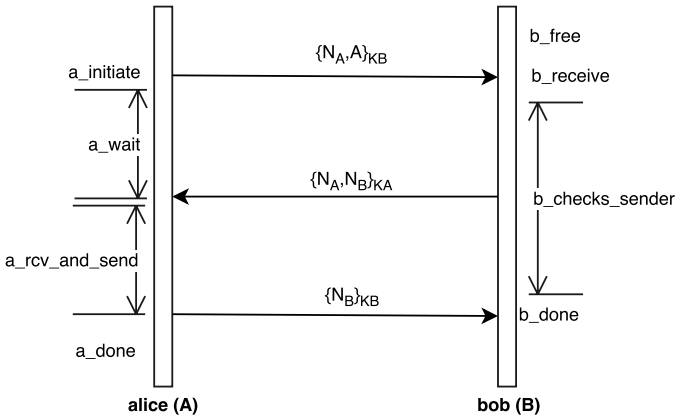

Figure 1: Model of Needham-Schroeder Protocol

3.1.2 Verification Specifications

- It is always the case that if Agent A initiates a session then both Agent A and B should reach a state where session is successfully established under the rules of the protocol.

The NuXMV code that models the protocol can be found here.

3.2 Otway-Rees Protocol

3.2.1 Model

The NuXMV model includes the following modules to represent the different components of Otway-Rees Protocol:

- `module alice` : To represent agent A
- `module bob` : To represent agent B
- `module server` : To represent server S

Each message packet traversing in the channel includes 3 fields encrypted using a certain key. The 3 fields can carry nonce, identifiers, keys or can be empty depending upon the stage of the protocol execution. The parameters to a module include two such packets one for sending, the other for receiving. Each of the parameters are allowed to hold certain values pertaining to the protocol execution. The parameters that each module take are :

- `send1`, `send2` , `sendkey`, `sendid`: To represent packet sent by an agent wherein `sendid` holds the value of the identifiers being sent in the message packet, `send1` holds the nonce and `send2` holds the a key or no key.
- `rcv1`, `rcv2` , `rcvkey`, `rcvid` : To represent packet received by an agent `rcvid` holds the value of the identifiers being received in the message packet, `rcv1` holds the nonce and `rcv2` holds the a key or no key.

Each module also have a set of states that refer to the stage in which the agent is in during the execution of the protocol :

- For agent A: {`a_initiate,a_wait,a_done`}
- For agent B: {`b_rcvA,b_sendA,b_rcvS,b_sendS,b_done`}
- For server S: {`s_idle,s_rcvB,s_sendB,s_done`}

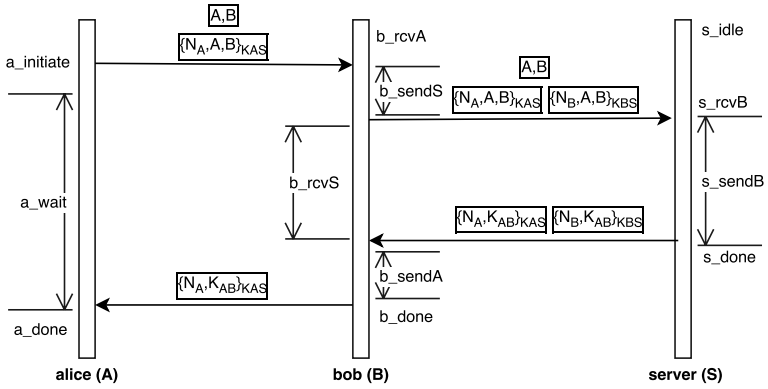

Figure 2: Model of Otway-Rees Protocol

3.2.2 Verification Specifications

- It is always the case that if Agent A initiates a session then both Agent A and B should reach a state where session is successfully established under the rules of the protocol.

The NuXMV code that models the protocol can be found here

3.3 Kerberos Protocol

3.3.1 Model

The NuXMV model includes the following modules to represent the different components of Kerberos Protocol:

- `module alice` : To represent agent A
- `module bob` : To represent agent B
- `module server` : To represent server S

Each message packet essentially consists of 5 fields namely, encryption key, session key, timestamp, lifetime and identifier. Each agent can send or receive 2 such packets which are referred to as "chunks". 2 chunks together form a `send_msg` or `rcv_msg`. The parameters that each module take are :

- `send_msg`: To represent packet sent by an agent/server which contains 2 chunks where each chunk has 5 fields as mentioned above.

- `rcv_msg` : To represent packet received by an agent/server which contains 2 chunks where each chunk has 5 fields as mentioned above.

Each module also have a set of states that refer to the stage in which the agent is in during the execution of the protocol :

- For agent A: {`a_idle,a_initiate_s,a_send_s,a_rcv_s,a_send_b,a_done`}
- For agent B: {`b_idle,b_rcv_a,b_send_a,b_done`}
- For server S: {`s_idle,s_rcv_a,s_send_a,s_done`}

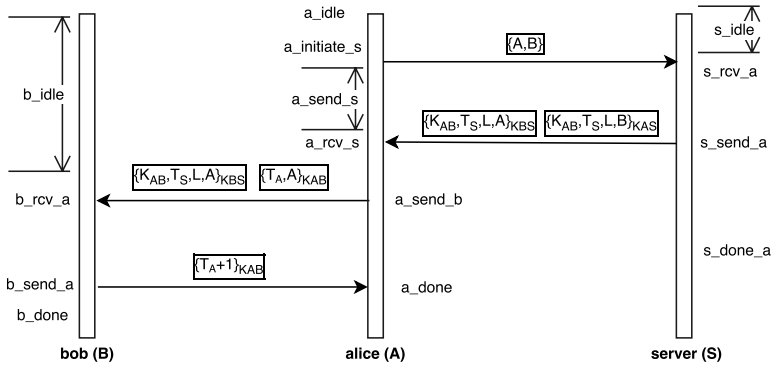

Figure 3: Model of Kerberos Protocol

Each agent has a set of counters to keep track of number sessions or interactions that are active. These counters are useful in formulation of the authentication and secrecy specifications.

- `a_begincount_b` : Counter in `alice` to keep track of number of sessions initiated by `alice` with `bob`
- `a_sendSK_b` : Counter in `alice` to keep track of number of session keys sent by `alice` to `bob`
- `b_endcount_a` : Counter in `alice` to keep track of number of sessions ended by `bob` with `alice`
- `b_rcvSK_a` : Counter in `alice` to keep track of number of session keys received by `bob` from `alice`

The timestamp in this model is assumed to be between 0 and 5. The lifetime is set to a constant value in order to abstract away an active session[**?**]. The major assumption in the design is only beginning of a session is modelled i.e when Agent A and Agent B to authenticate each other. A running session that continues after authentication is assumed to be a secure session and hence, not designed.

3.3.2 Verification Specifications

- Authentication Specification : It is always the case that the number of sessions started by Agent A with B $\geq$ Number of sessions Agent B accepts with A.
- Secrecy Specification : Number of session keys shared by Agent A with B is $\geq$ Number of session keys Agent B accepts from A

The NuXMV code that models the protocol can be found here

4 Implementation

All the models are drawn and verified by use of version 1.1.0 of NuXMV. Each protocol is first modelled in absence of intruder to represent the functioning of protocol. Each model also undergoes interactions among agents that are not specified in the protocol execution, however they do not lead to the success state. Presence of unacceptable states validates that the model represents universal states and not just the desirable states.

Each protocol is then injected with an intruder that has capability of eavesdropping, generating new message components, removing message component, and impersonation. The model checking of the protocol is then carried out in presence of an intruder to verify whether the intruder gets hold of some message component that is crucial to the protocol execution and renders the protocol to be attack-prone.

Presence of attack is determined by the counterexamples that are generated by the mode checker when the protocol is tested against certain security specification. The security specification fails to hold in case of attack to protocol and hence, generates a counterexample representing the series of actions that result in an attack to the protocol.

5 Test and Results

5.1 Needham-Schroeder Public Key Protocol

The NuXMV code that models the attack on the protocol can be found here

5.1.1 Attack[?]

An attack in Needham-Schroeder Public Key Protocol suffers an attack due to intruder impersonation. If the intruder I manages to establish a connection with honest agent A to impersonate as B, denoted as $I(B)$. I is able to extract the secret from the packets that are sent to it from A and A, at different stages of the protocol, sends secrets N_A and N_B to I thinking it to be an honest agent. The attack is represented in Figure 4.

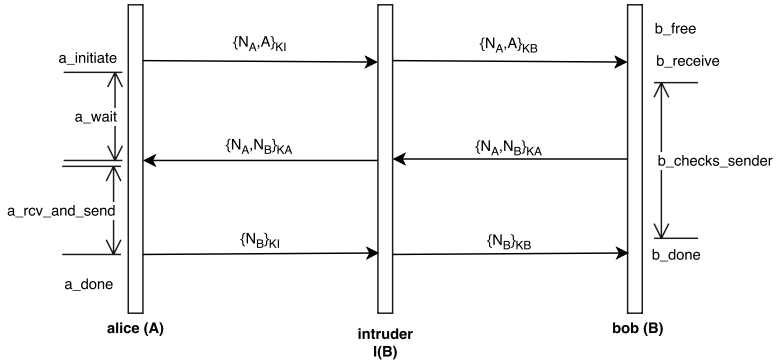

Figure 4: Attack on Needham-Schroeder Public Key Protocol

5.1.2 Violated Specifications

- If Agent A initiates a session then both Agent A and B reach a state where session is successfully established and the intruder I does not have the secret information from A and B.

Counterexample generated by the model-checker that signifies attack can be found here.

5.1.3 Suggested Resolution

The attack detected is of type impersonation where the intruder pretends to be Agent A with Agent B. This attack can be resolved if Agent B sends its identifier during its send stage. Since, Agent A is aware of its communication with intruder I assuming it to be honest, presence of identifier from the actual Agent B would help A distinguish between the honest and the pretentious agent B.

5.2 Otway-Rees Protocol

The NuXMV code that models the attack on the protocol can be found here

5.2.1 Attack[?]

An attack in Otway-Rees Protocol suffers a man-in-middle attack. In the attack, the intruder I manages to impersonate as B, denoted as $I(B)$ and establishes a connection with A. In the last step of the protocol execution, $I(B)$ sends back the same packet to A that it received in the first step, which now A mistakes the identifiers to be the new session key. Now both A and $I(B)$ have the same session key i.e $\{A, B\}$ which is used for securing the subsequent message exchanges. The attack is represented in Figure 5.

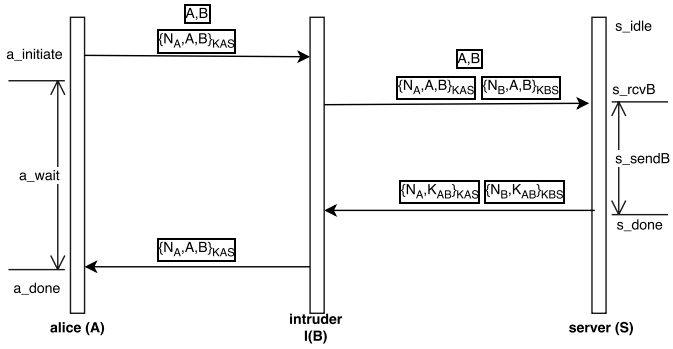

Figure 5: Attack on Otway-Rees Protocol

5.2.2 Violated Specifications

- If Agent A initiates a session then both Agent A and I(B) reach a state where session is successfully established and the intruder I(B)

does NOT share a common key with A

Counterexample generated by the model-checker that signifies attack can be found here.

5.2.3 Suggested Resolution

The attack detected is of man-in-the-middle type of attack, This attack can be resolved if in the last stage of the protocol execution, the protocol is modified to make Agent B send two message packets to Agent A where the first packet contains the encryption key for the second packet. In such scenario, the intruder I(B) cannot gain benefit re-sending the same packet because a different encryption key is expected by Agent A.

5.3 Kerberos Protocol

The NuXMV code that models the attack on the protocol can be found here

5.3.1 Attack[?]

An attack in Kerberos Protocol is termed as replay attack. In this attack, the intruder $I(A)$ manages to intercept the message while A sends the session key to B and connects with B. Now, B assumes that it has established two sessions with A and it continues towards completing the protocol execution. After the protocol reaches the completion step, B keeps sending the secret messages in both session, without realizing that one of the session is with a intruder. The attack is represented in Figure 6.

5.3.2 Violated Specifications

- Authentication : It is always the case that the number of sessions started by Agent I with B >= Number of sessions Agent B accepts with I.
- Secrecy : Number of session keys shared by Agent I with B >= Number of session keys Agent I accepts from A.

The violations suggest that the Agent B establishes connection with intruder I and executes exchange of messages similar to those with Agent A and thus, the intruder is able get Agent B into confidence.

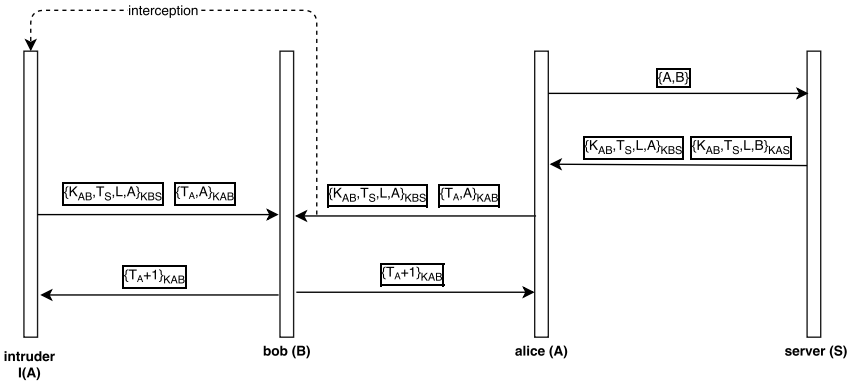

Figure 6: Attack on Kerberos Protocol

5.3.3 Suggested Resolution

The attack detected is a replay-attack wherein the imposter sets up a duplicate session with agent B. This attack can be resolved if agent B stores all the incoming authenticators from all the live sessions to detect duplicate ones to avoid replay-attack.

6 Conclusion

This project illustrates automatic attack detection in protocols like Needham-Schroeder, Otway-Rees and Kerberos by modelling the protocols in the NuXMVmodel checker. The intention of this project is to utilize the formal methods to verify a critical system such as security protocol that form the basis of secure transmission of information. Detection of possible attacks and loopholes in the existing protocols is essential for development of protocols that employ methods that can secure the information and can avoid possible attacks.

Appendix A NuXMV Program Code

A.1 Needham-Schroeder Protocol

Protocol Model
Protocol + Intruder Model

A.2 Otway-Rees Protocol

Protocol Model
Protocol + Intruder Model

A.3 Kerberos Protocol

Protocol Model
Protocol + Intruder Model